

**ÓBUDAI EGYETEM**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Bánki Donát Gépész és Biztonságtechnikai Mérnöki Kar**

**Szabaduló szoba vezérlő szoftver tervezése**

**OE-BGK** Hallgató neve: **Kovács Árpád**

**2021.** Neptun kód: BPJZ56

**Tartalomjegyzék**

[Jelmagyarázat 4](#_30j0zll)

[Köszönetnyilvánítás 5](#_1fob9te)

[1.](#_3znysh7) Bevezetés 6

[2.](#_2et92p0) Szakirodalom elemző feldolgozása 7

[2.1.](#_tyjcwt) Firebase 7

[2.1.1.](#_3dy6vkm) A modellek 8

[2.1.2.](#_1t3h5sf) Nézetek 9

[2.1.3.](#_4d34og8) Kezelő 9

[2.2.](#_2s8eyo1) Babel fordító könyvtár 9

[2.3.](#_17dp8vu) A Blockly könyvtár 9

[2.4.](#_3rdcrjn) Raspberry PI 400 9

[2.5.](#_26in1rg) Manjaro ARM 10

[2.6.](#_lnxbz9) NPM/Node.JS 10

[2.7.](#_35nkun2) Github 10

[2.8.](#_1ksv4uv) ESP32 10

[2.9.](#_44sinio) RFID olvasó 11

[2.10.](#_2jxsxqh) OLED kijelző 12

[2.11.](#_z337ya) Fejlesztőkörnyezet - Visual Studio Code 12

[3.](#_3j2qqm3) Saját munka 14

[3.1.](#_1y810tw) Firebase modell: 15

[3.2.](#_4i7ojhp) Hálózatkezelés: 21

[3.3.](#_2xcytpi) Github Action: 21

[3.4.](#_1ci93xb) Vue rész:. 22

[3.5.](#_3whwml4) Szabaduló szoba programok programozása: 24

[3.6.](#_2bn6wsx) Programok futattása: 32

[3.7.](#_qsh70q) Kamerák kezelése: 35

[3.8.](#_3as4poj) Az eszközök kezelése: 35

[3.9.](#_1pxezwc) Az ESP32 programja: 50

[3.10.](#_49x2ik5) A regisztráció menete: 52

[3.11.](#_2p2csry) Az eszköz indulása: 54

[3.12.](#_147n2zr) Az eszköz bejelentkezése: 56

[3.13.](#_3o7alnk) Relé üzemmód: 56

[3.14.](#_23ckvvd) Bemenet üzemmód: 57

[3.15.](#_ihv636) Megjelenítő üzemmód 58

[3.16.](#_32hioqz) RFID üzemmód 58

[3.17.](#_1hmsyys) Raspberry PI beállítása:. 60

[3.18.](#_41mghml) A szoftver futattása Raspberry PI-n:. 60

[4.](#_2grqrue) Összefoglalás 62

[5.](#_vx1227) Idegen nyelvű tartalmi összefoglaló 63

[6.](#_3fwokq0) Irodalomjegyzék 64

[7.](#_1v1yuxt) Mellékletek 65

[Nyomtatott melléklet 65](#_4f1mdlm)

# Jelmagyarázat

|  |  |
| --- | --- |
| **Jelzés/Rövidítés** | **Jelmagyarázat** |
| **Firebase** | A google által fejlesztett Backend as Service könyvtár |
| **Axios** | Egy valós idejű támogatást biztosíŧó könyvtár |
| **JavaScript** | Egy böngészőben futó szkript nyelv |
| **HTML** | HyperText Markup Language |
| **CSS** | Cascading Style Sheets |
| **Tailwind** | CSS könyvtár |
| **Bootstrap** | CSS könyvtár |
| **RFID** | Radio-frequency identification |
| **OLED** | Organikus fénykibocsájtó dióda |
| **Blockly** | Google által fejlesztett grafikus programozó fejlesztő könyvtár |

# Köszönetnyilvánítás

Köszönettel tartozom Dr. Simon János tanár úrnak, a mentorom volt a feladat megvalósítása során. Amikor felötlött bennem a szakdolgozat témája, hozzá fordultam és a véleményét kértem róla. Miután elmagyaráztam, hogy mit is szeretnék akkor alaposan átbeszéltük a részleteket és még bővítettük a már amúgy is vaskos teendők listáját. Megbeszéltük, hogy az elkészült projektnek minden olyan eszközön működnie kell, amelyen a vezető modern böngészők valamely változata fut.

A Rehák kollégium volt és jelenlegi lakóinak.

Valamint köszönettel tartozom még a családomnak, akik nagyban támogattak és hozzájárultak ahhoz, hogy eljuthattam idáig és megírhatam a szakdolgozatomat.

1. **Bevezetés**

A feladat egy olyan szabaduló szoba vezérlés létrehozása, mellyel akár a programot távolról is lehessen futattni. A programnak támogatnia kell az ESP32 mikrovezérlőt, valamint tartalmazza, ábrázolja, és engedélyezze a szobák szoftverének szerkesztését.

A feladatot úgy kell megoldani, hogy Raspberry PI mikroszámítógépen is futtatható legyen.

A dokumentum célja a diplomamunka dokumentálása és leírása, a szakdolgozat fő célkitűzése volt, egy olyan szoftver készítése, mely a szabaduló szobáknak a létrehozásában és menedzselésében segít. A szoftvernek támogatnia kell, új szabaduló szobák létrehozását, ESP32 mikrovezérlő szoftverének vezérlését HTTP-protokolon, valamint a szobát fenntartó személyzet segítését, a csapatok nyomon követését, és nem utolsósorban a szobák programozását.

A feladat eredménye egy olyan alkalmazás létrejötte, mely engedi a felhasználónak, hogy távolról is dolgozhasson.

1. **Szakirodalom elemző feldolgozása**

A szakdolgozat megvalósításához a következő technológiákat használtam fel:

* HTML5
* JavaScript
* Vue.JS
* Firebase
* Axios/Ajax
* Node.js/NPM
* CSS
* Material CSS
* Arduino IDE
* ESP32

A feladathoz az esp32 típusú mikrovezérlőt lett választva, a feladatot úgy oldottam meg, hogy a program, egy közös csatornán keresztül kommunikál a szerverként üzemelő számítógéppel, és ez úgy lett megvalósítva, hogy azonos hálózathoz, van kötve az összes kliensként dolgozó Raspberry PI 400.

A feladat megvalósításához Firebase keretrendszert használtam fel.

* + 1. **Firebase**

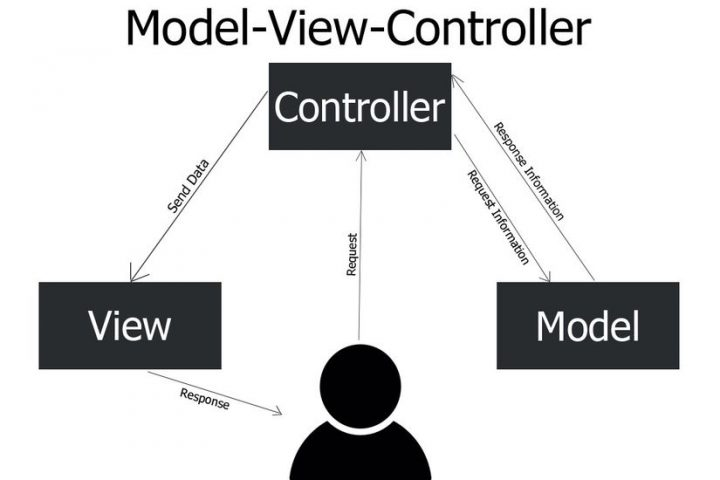
A Firebase keretrendszer egy Google fejlesztése alatt áló adatbázis és hozzá tartozó rendszerek között használhatóak valós idejű adatbázisok, Collectionok, valamint, Statikus adattárolás, Fájl tárolás, illetve Hostingolásra is. A firebase egy úgynevezett Backend as Service alapú struktúrát követ mely segítségével az ilyen alkalmazások létrehozása, skálázott és a megfelelő használatban ingyenes is.

A firebase adatmodellek a MongoDB szerűen Node-okba tárolják el az adatokat. S ehhez megfelőlően használtam fel én is. Én az adatbázissal való összekötésre, az ún MVC modellt használtam, mely alapján az adatot tároljuk módosítjuk, annak az adatját Controlleren kezeljük, illetve megjelenítjük.

Az adatbázis ún nem relációs adatbázisok on keresztül kommunikál a Javascript felület illetve a mikrokontroller.

A Firebase kezeli a felhasználók authentikációját is.

.



*ábra SEQ ábra \\* ARABIC 1 MVC modell*

Ezek végeztével elkezdtem kidolgozni a feladathoz szükséges csomópontok elemeit, később feltelepítettem a Firebase könytárait a keretrendszerbe, e program segítségével elkezdtem elkészíteni felhasználói felületet, mely Vue.JS Javascript támogatásával üzemel.

Vue 3.0 segítségével elkezdtem az egy oldalas alkalmazás létrehozzását, mely csak Ajax használatával kommunikál a rendszerrel, a többit egy előre fordított Javascript-ten keresztül tölti be.

A modulokat létrehoztam, majd elkezdtem a weboldalhoz tartozó weboldalakat létrehozni. Ezek egy Vue.js könyvtárban találhatóak, melyeknek direkt kommunikációval rendelkeznek a weboldalhoz. Ezt egy úgy nevezett[[1]](#footnote-2) Axios.JS csomagon keresztül intézi.

Mivel a feladatmegvalósításhoz MVC-t használam fel ezért szükséges elmagyaráznom, mit tartalmazz magában.

* + - 1. ***A modellek***

A modellek amik tárolják adatok struktúráját, amelyek segítségével kommunikálunk az adatbázzissal, valamint ezáltal az adatokat írjuk bele.

Erre példa amit használtam programok tárolásánál. Majd ennek a modellhez szükséges kidolgozni az adatbázis felöli kezelő felületét is.

* + - 1. ***Nézetek***

Ebben a projektben ezek az útvonalaknak felelnek meg amit a rendszer felhasznál az előre fordított javascript fileoknál. Normál esetben a lekért útvonalat a szerver maga abban a pillanatban generálja le, és adja át a böngészőnek.

* + - 1. ***Kezelő***

Itt lehetséges a modellek viselkedését meghatározni eseményekre, pl. a lekért modellhez rakjon össze útvonalat és jelenítse meg. Ebben az esetben az Inertia.JS felel a nézet és kontroller közötti kommunikációért.

* + 1. **Babel fordító könyvtár**

Ez a könyvár lehetővé teszi újabb alkalmazások pl. (Vue, React, vagy Angular) típusú alkalmazások felhasználást régebbi böngészőre, valamint a kódot optimalizálja. Az így létrehozott Stílus és Javascript fájl nem több pár megabytenál ezzel levéve a súlyos fileok mozgatását a szerverről.

Ezt használja fel a WebPack technológiának is.

* + 1. **A Blockly könyvtár**

Google által fejlesztett grafikus programozási program, mely képes több nyelvre is lefordítani a lehelyezett elemeket, támogatott a php, javascript, go stb.

Az itt felhasznált része annyiban merül ki, hogy a felhasználó ebben rakja össze az ő általa írt programokat, és majd ezeket lementi és lefutattja a kezelő felület.

Ehhez szükséges volt egy külön konzolt létrehozni, mely lehetővé teszi a felhasználóknak a blockly által támogatott blockok futattását, valamint azok reakcióját a valós rendszerrel.

* + 1. **Raspberry PI 400**

A szolgáltatás futattására én a Raspberry PI 400-as mikroszámítógépet használtam fel. Ezt használja a felhasználó az eszköz kezelésére. A Raspberry Pi egy bankkártya méretű, egyetlen áramköri lapra/kártyára integrált BCM2835 alapú egykártyás számítógép, amelyet az Egyesült Királyságban fejlesztettek oktatási célokra. Az eredeti két változat (A és B) kiadása óta már több továbbfejlesztése is kiadásra került. A hivatalosan ajánlott operációs rendszer a laphoz a Raspbian, ami a Debian Linux kifejezetten Raspberry Pi-re optimalizált változata. A legfrissebb modellje a 3. generációs Raspberry Pi 3, de időközben kiadtak még kisebb méretű Zero változatokat is.

* + 1. **Manjaro ARM**

Manjaro ARM olyan operációs rendszer, mely a Raspberry PI 400-on fut. A Manjaro egy Arch típusú Linux disztrubició.

* + 1. **NPM/Node.JS**

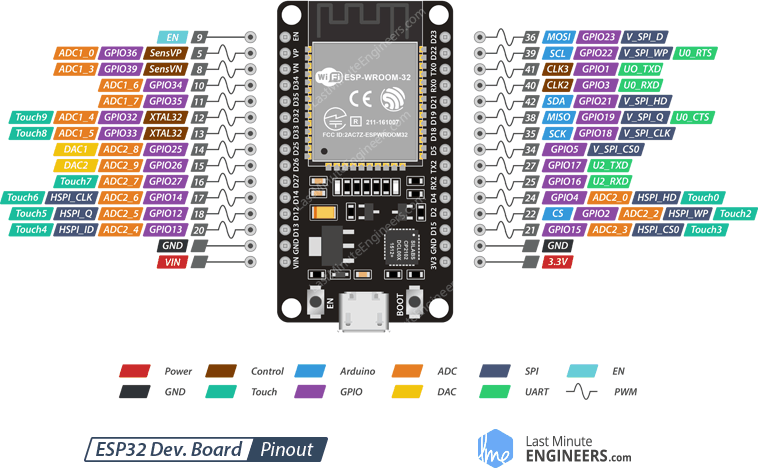
A rendszer az npm könyvártkezelő segítségevel építí fel magát.

* + 1. **Github**

A szoftver szervessen támaszkodik Github adattárolóra, egy rész a fejleszés itt történt, valamint a Github Action automata feltöltése a Firebase hosting-ra is itt történt, valamint az ESP32 is innen tölti le a legújabb verzióját az ESP32-nek.

* + 1. **ESP32**

Az esp32 az espressif által tervezett és gyártott mikrokontroller. Ezt használtam fel a feladat megoldásához, ez az eszköz kommunikál a Raspberry Pi-vel a Firebase API csatornáján és állítja be a szoftvert.

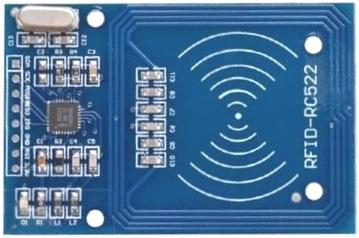
[[2]](#footnote-3)

*ábra 2 ESP32*

Az ESP32 támogat többféle protokollt, egyelőre 4 dolog lett kidolgozva, képes relé üzemmódban dolgozni, érzékeli azt és elküldi az RFID kártya ID-jét a szervernek, valamint képes bemeneti módban is dolgozni azáltal, hogy a megjelölt (25) portot rövidre zárjuk a földdel. Nem utolsósorban képes az eszköz megjelenítő eszközként is dolgozni amennyiben egy OLED kijelzőt kötünk rá.

* + 1. **RFID olvasó**

Az RFID egy rádiós technológián alapúló azonosító eszköz. Különféle eszközök használhatnak RFID tag-eket. S az ESP32 is képes ilyenek olvasására az MFRC522 lappal.

[[3]](#footnote-4)[[4]](#footnote-5)

*ábra 3 MFRC522 RFID olvasó*

* + 1. **OLED kijelző**

Az OLED kijelző egy grafikus megjelenítő eszköz, melynek segítségével szöveget írhatunk ki az ESP32 eszközről.

[[5]](#footnote-6)

*ábra 4 OLED kijelző*

* + 1. **Fejlesztőkörnyezet - Visual Studio Code**

Ez egy a Microsoft által fejlesztett IDE, amely integrált fejlesztő környezetet jelent. A Visual Studio Code talán napjaink egyik legfejlettebb IDE-je, minden eszköz a fejlesztő rendelkezésére áll.

Található benne:

* Debugger
* Automatikus kód kiegészítés
* Különféle fejlesztői tesztek
* Szintaxis kiemelés
* Automatikus kódformázás

A Visual Studio Code azért is rendkívül jó IDE, mert rengeteg programozási nyelvet támogat. Csak, hogy néhányat említsek:

* JavaScript
* Node.js
* HTML
* CSS
* C#
* C++
* F#
* CoffeScript
* TypeScript

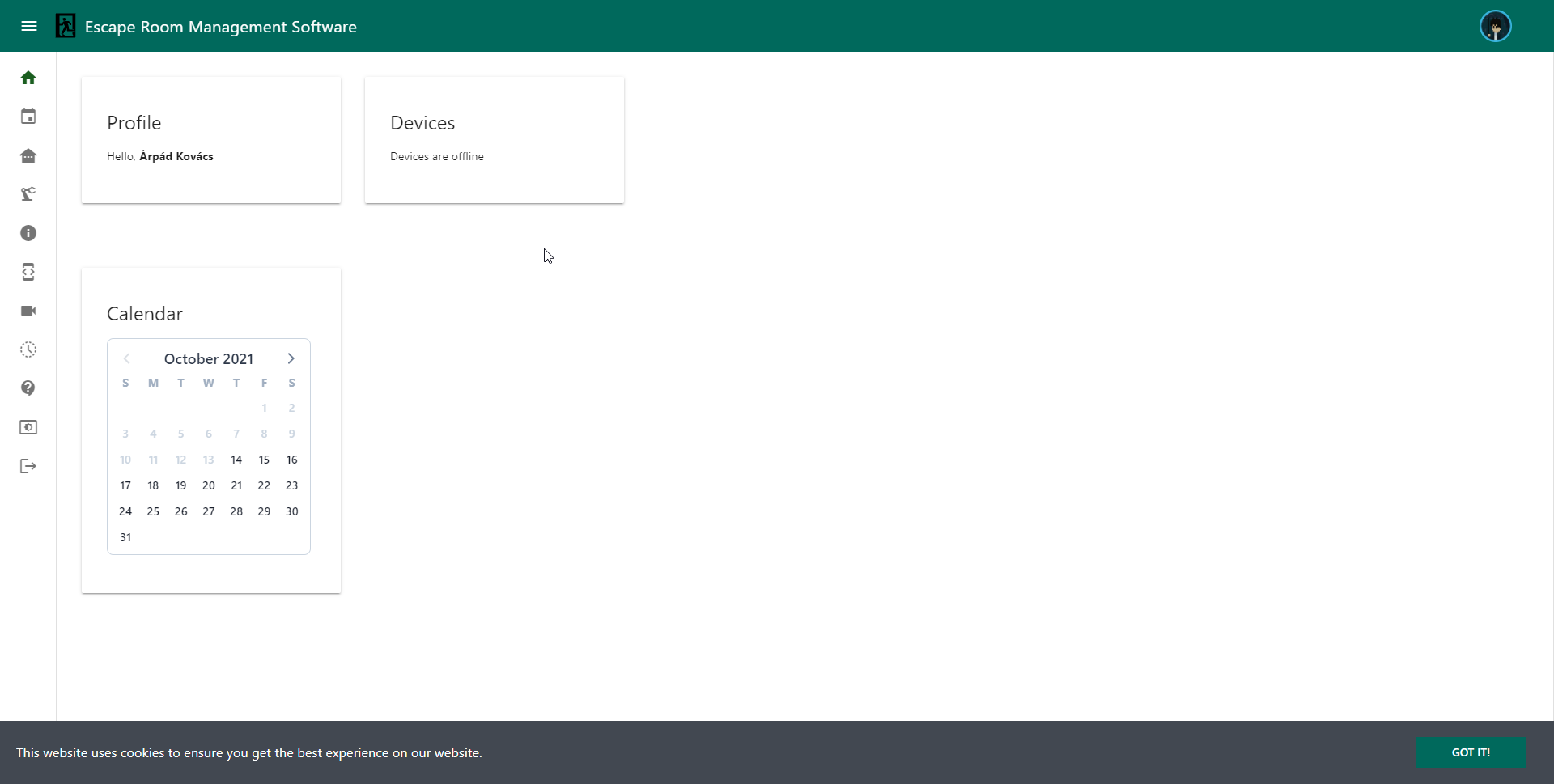
A szakdolgozat megvalósításánál ebben készítettem a JavaScript, Node.js, HTML és CSS részeket.

1. **Saját munka**

Ebben a fejezetben fogom ismertetni a szakdolgozat megvalósítását. A szakdolgozat két részből áll. Az első maga a szerver elkészítése, a második pedig a kliens alkalmazás elkészítése. Mivel mind a két rész igen terjedelmes és összetett, ezért csak a fontosabb részeket fogom ismertetni. A többi rész megtalálható forráskóddal együtt a CD mellékletben. Valamint röviden bemutatom a fejlesztésre használt eszközöket is. Először is talán kezdjük a rendszer blokk vázlatával:

|  |
| --- |
| Valós idejű adatbázis  Firebase szerver Kliens  Wifi  Router  ESP32  Firebase  Vue felhasználói felület  ESP32 |

*ábra 5 A rendszer blokkvázlata*



*ábra 6 Az alkalmazás cím oldala*

A Firebase keretrendszert választottam majd, elkezdtem létrehozni a kezdő struktúrát, kezdve a felhasználók struktúrújával.

Ehhez a Vue 3.0 starterkit-jét telepítettem, mely támogatja a CRFS védelmet, valamint a botok elleni védelmet is, ennek a telepítő kódja:

|  |
| --- |
| npm init obudai\_diplomamunka  npm install vue  npm run dev |

* + 1. **Firebase modell:**

Ez alapján kezdtem el kidolgozni az MVC modellt. Meg kell említeni, hogy a rendszer elsődlegesen a firebase adatbázis kommunikációra támaszkodik adatkommunikációt használ, míg a programkínézetek előre vannak fordítva az app.js file-ba, a Babel könyvtár segítségével.

Modellekel kezdtem.

A firebase hozzáférés kezelése:.

|  |
| --- |
| {    "rules":{      //".read": true,      "users": {        "$uid": {          ".write": "$uid === auth.uid",           ".read": "$uid === auth.uid",             "rooms":             {               "$roomid":{        "devices":        {          "$devid":{          ".write": true,          ".read": true,          }        }             }               }        }      }  }  } |

A szobák tárolása

A szobákat a felhasználók alatt egy AuthID található /rooms Nodeban tárolja a rendszer.

Szobák tartalmazzák a legfontosabb adatokat, eszközöket, illetve az ehhez tartozó beállításokat is.

A szoba tárolja az eszközöket, szenzorokat, előző szabadulási időket, csapatokkal, bejegyzett időpontokat, kapcsolódási pontot (Wifi), illetve programokat, s ezek beállítását is.

Az eszközök kezelése

Ezt a hozzáférést egy ún. REST API-n keresztül történik. S majd ezt az API-t hozzáférésről a .json fájl kiterjesztésével lehet lekérdezni, s frissíteni.

Mely azt jelenti, hogy az alkalmazásnak HTTP kéréseken keresztül lehet elérni.

Ez azt jelenti, hogy ugye a biztonsági táblázat meghatározza az eszközöknek a hozzáférést, s hogy adatmódosításának biztonsági rést alkott, ha valaki tudja, az eszköz ID-jét. ezért éreztem szükségesnek még a szobának az ID-jébe is beágyazni, s ezáltal lehet csak az eszköz állapotát módosítani.

Azért választottam a Rest API-t nem az MQTT protokoll mert egyszerűbbnek éreztem az implementálását többféle eszközre, ezzel a megoldással akár, PLC-t is lehet kapcsolgatni, mivel egy JSON kódolva kapja meg az adatokat így egyszerűbb ennek a lekérdezési formátuma.

Az eszközök az esp32-k programozhatóak weboldalról is, ezt egy ún. Serial.JS könyvtár teszi lehetővé mely segítségével a Serial porton keresztül az alkalmazás ellenőrzi, hogy a program aktív, valamint beállítja a kérdezési útvonalat pl.

|  |
| --- |
| https://escaperoom-b4ae9-default-rtdb.europe-west1.firebasedatabase.app/users/6ZMB5XNK1gd2PsJA8ae8vJs9jkP2/rooms/-MibGQVxewjkAo66aY67/devices/-Mieerw2m\_Ess8--VHdF.json |

Majd ennek végeztével, egy HTTP lekérdezést csinált 2 másodpercenként. Érdemes megemlíteni a Firebase korlátjait ebben az esetben. Mivel a Firebase és én az ingyenes csomagot használtam, ezért 5 gb adatforgalom van megengedve havonta a valós idejű adatbázisra.

Itt frissíti a bázisban található időt közvetlenül lekérdezés után, sajnos ezt az eszköznek kell végrehajtania, mivel a bázis csak a prémiumnál engedélyezi a “belső szkriptek futtatását”.

Így az eszköznek kell frissítenie az órát mely eltérhet a Belgiumi időzónától ezért szükséges az ESP32 még, mindennek előtt egy NTP szerverrel összehangolni, mely segítségével ilyen probléma nem fordul elő.

A programok tárolása:.

A programokat a szovtver a szoba Room - nodeban tárolja a programs node alatt.

itt is a felhasználók korlátlanul tudnak hozzáadni a programokat, amelyeket a Blockly felülettel hoznak létre. Azért jó ez a “hierarchikus” elrendezés mivel így az adatok minimálisan keverednek, de viszont létrejöhetnek duplikátumok ugyanazon adatokból.

A programokat mentés után, mely automatikusan történik változtatás után, a program BASE64-be lekódolja és ez kerül bele a “program\_xml” mezőbe. Ezért arra volt szükség, hogy a szoftver ne tároljon két másolatott ugyanarról a szoftver, egy Javascript alapút, s egy XML alapú.

E kódolás változtatható, mivel a program tárolja azt, hogy mivel lett kódolva, s ez alapján vissza is lehessen oldani, de ehhez a felhasználó nem fér hozzá.

A programokat lehet biztonsági menteni is, ilyenkor a program a teljes Node tartalmát, egy JSON file ba kiírja és letöltetti a felhasználóval.

Ezt a fájlt ugyanígy vissza is lehet tölteni.

Az aktív programnak kiválasztását az Active\_program cella tárolja, itt található blokkba a felhasználó kijelöli az Aktív programot, s ennek a programnak az ID-jét, a szoftver beírja a  cellába.

Ezek végeztével elkezdtem kidolgozni a nézeteket is. Ezeknek a használatához igénybe vettem a Vue.JS könyvtárat. Majd párhuzamosan kidolgoztam az útvonalat (Route-t) melyeket VueRoute modullal hoztam létre és a hozzá tartozó nézetet. Mivel a cél az egy oldalas alkalmazás készítése volt, ezért a rendszer technológia Stackje megfelel már az egy oldalas feltételnek.

Ezt a modult feltelepítve létrehoztam a következő rendszert.

A vue route javascript file tartalma:

|  |
| --- |
| import Vue from 'vue';  import VueRouter  from 'vue-router';  import {FirebaseAuth} from '../firebase';  Vue.use(VueRouter);  import Index from "../components/Index";  import Error404 from "../components/Error404";  import AccountIndex from "../components/account/Index";  import AccountInfo from "../components/account/AccountInfo";  import AccountLogin from "../components/account/Login";  import Home from "../components/Home";  import Info from "../components/Info";  import Rooms from "../components/Rooms";  import Room from "../components/Room";  import Devices from "../components/Devices";  import Device from "../components/Device";  import Programs from "../components/Programs";  import Program from "../components/Program";  import Lobby from "../components/Lobby";  import Camera from "../components/Camera";  import Cameras from "../components/Cameras";  import PreviousRuns from "@/components/PreviousRuns";  import PreviousRun from "@/components/PreviousRun";  import Events from "@/components/Events";  import Event from "@/components/Event";  import Support from "@/components/Support";    const router = new VueRouter ({      mode: 'history',      base: '/',      routes: [          {              path: '\*',              name: 'error404',              component: Error404,          },          {              path: '/',              component: Index,              meta: {                  requiresAuth: true              },              children: [                  {                      path: 'home',                      name: 'home',                      component: Home,                  },                  {                      path: 'rooms',                      name: 'rooms',                      component: Rooms,                  },                  {                      path: 'devices',                      name: 'devices',                      component: Devices,                  },                  {                      path: 'programs',                      name: 'programs',                      component: Programs,                  },                  //                  {                      path: 'cameras',                      name: 'cameras',                      component: Cameras,                  },                  {                      path: 'pruns',                      name: 'pruns',                      component: PreviousRuns,                  },                  {                      path: 'room',                      name: 'room',                      component: Room,                  },                  {                      path: 'info',                      name: 'info',                      component: Info,                  },                  {                      path: 'logout',                      name: 'logout',                      component: Info,                  },                  {                      path: 'events',                      name: 'event',                      component: Events,                  },                  {                      path: 'support',                      name: 'support',                      component: Support,                  },                  {                      path: '/room/:rid',                      component: Room,                  },                  {                      path: '/room/:rid/device/:did',                      component: Device,                  },                  {                      path: '/room/:rid/program/:pid',                      component: Program,                  },                  {                      path: '/room/:rid/event/:eid',                      component: Event,                  },                  {                      path: '/room/:rid/lobby',                      component: Lobby,                  },                    {                      path: '/room/:rid/camera/:cid',                      component: Camera                  },                  {                      path: '/room/:rid/prun/:prid',                      component: PreviousRun                  },              ]          },          {              path: '/account',              component: AccountIndex,              children: [                  {                      path: '',                      name: 'account-info',                      component: AccountInfo,                  },                  {                      path: 'login',                      name: 'account-login',                      component: AccountLogin,                  }              ]          }      ]  });  router.beforeEach((to, from, next) => {      let currentUser = FirebaseAuth.currentUser;      let requiresAuth = to.matched.some(record => record.meta.requiresAuth);      if(requiresAuth && !currentUser) next('/account/login');      else next();  })  export default router; |

Ha a felhasználó nem jelentkezett be automatikus visszautasítja a rendszer a bejelentkezési képernyőre.

* + 1. **Hálózatkezelés:**

A hálózati összekötésre a REST API-t használja a rendszer. S azonnal frissíti a rendszer magát.

Mivel a böngészők automatikusan lekezelik a https kérését, de azonban az ESP32-nek szükséges az Igazolás az eszközön, hogy rajta legyen. Ezt szükséges volt egy böngészőből importálni. S ezt a fájlt az ESP32 Spiffs adatárolójába lementeni.

* + 1. **Github Action:**

A github action egy speciális szkript amely lehetővé teszi, 1 bizonyos műveletek végrehajtását, időközönként, vagy valamilyen akcióra reagáljon. Ebben az esetben Push relációra automatikusan lefordítja ezáltal ellenőrizve, hogy van-e benne hiba és ha nincs benne hiba akkor, deploy-olja a weboldalt a következő címre:.

<https://escaperoom-b4ae9.web.app/>

|  |
| --- |
| name: Deploy to Firebase Hosting on merge  'on':    push:      branches:        - main  jobs:    build\_and\_deploy:      runs-on: ubuntu-latest      steps:          - uses: actions/checkout@v2        - run: echo '${{ secrets.CREDENTIALJS }}'  > ./src/firebase/credentials.js          - run: npm install && npm run build        - uses: FirebaseExtended/action-hosting-deploy@v0          with:            repoToken: '${{ secrets.GITHUB\_TOKEN }}'            firebaseServiceAccount: '${{ secrets.FIREBASE\_SERVICE\_ACCOUNT\_ESCAPEROOM\_B4AE9 }}'            channelId: live            projectId: escaperoom-b4ae9 |

Az ESP32 felöli autoupdate:.

Ha Pusholás történik akkor a Github egy virtuális gép segítségével automatikusan lefordítja s feltölti az új bin fileoka.

* + 1. **Vue rész:.**

A Vue kinézeti fileok a következő képen épülnek fel:

|  |
| --- |
| <template>  Itt található a Vue elemekből összeállított modell (Html)  </template/>  <script>  Itt funckionalitást rakjuk össze.(JS)  export default {  components: {  },  data: ()=>  {  Itt találhatóak program változói,  },  methods:  {  program()  },  mounted: mounted()  {  Meghíváskor lefutó szkript  }  }  </script>  <style>  Míg a stilusban kinézeti css-t állítjuk be. (CSS)  </style> |

A felhasznált szkriptek a következő képen vannak eltárolva:

* Komponensek: több azonos modult felhasználó elemeket tárolok, itt vannak Vue eredeti filejai is pl. Checkbox.vue.
* Az oldalak: Létre van hozva egy Components nevű mappa ott találhatóak Vue oldalak, ami komponensekből áll.
* Rendszert konfiguráló beállítások: Ezek azok a fileok amik meghatározzák az alábbi fileok működését, pl. Itt van meghatározva, hogy az elemek Tailwind css-t használnak.

A program kiinduló pontja:.

|  |
| --- |
| import Vue from 'vue';  import router from './router';  import ForkeMeOnGithub from 'fork-me-on-github-vue';  import {IconsPlugin } from 'bootstrap-vue';  import BootstrapVue from "bootstrap-vue";  import VueNoty from 'vuejs-noty';  import VCalendar from 'v-calendar';  import AudioRecorder from 'vue-audio-recorder';  import VueMaterial from 'vue-material'; // TODO: import only needed component, not all  import 'vue-material/dist/vue-material.min.css';  Vue.use(VueMaterial);  //Vue.use(ForkeMeOnGithub);    Vue.use(AudioRecorder);  Vue.use(BootstrapVue);  import "bootstrap/dist/css/bootstrap.css";  import "bootstrap-vue/dist/bootstrap-vue.css";  Vue.use(IconsPlugin);  Vue.use(VCalendar, {})  Vue.use(VueNoty, {      timeout: 4000,      progressBar: true,      layout: 'topRight'  });    import App from './App.vue';  let app = null;  Vue.config.productionTip = false;  import { getAuth, onAuthStateChanged } from "firebase/auth";  const auth = getAuth();  onAuthStateChanged(auth, (user) => {      if (!app) {          app = new Vue({              router,              render: h => h(App)          });          //app.use(BootstrapIconsPlugin);          app.$mount('#app');      }  }); |

* + 1. **Szabaduló szoba programok programozása:**

A programnak szükséges támogatnia a Blockly könyvtárat, melynek fő használata program.vue fileon belül található. A blockly segítségével grafikusan lehet programozni.

S itt ez a szkript akkor hajtódik le amennyiben a program a szoba lobby fázisba kerül a Blockly lefordítja az eltárolt Base64-be letárolt xml-t lefordítja Javascript formátumba.

S A startra egy ún. Sandbox-ba rakja.

|  |
| --- |
| <template>  <div class="center">    <h2>Program ~ {{program.program\_name}}</h2>   <md-field>        <label>Program name:</label>        <md-input @change="namechange" v-model="program.program\_name"></md-input>      </md-field>  <div id="blocklyDiv"></div>  <Blocks :devices="devices" />  <md-button class="md-raised md-primary" @click="duplicateprogram()">Duplicate program</md-button>  <md-button class="md-raised md-secondary" @click="get\_config()">Download Program</md-button>    <md-field>  <md-button class="md-raised md-secondary" @click="showDeleteDialog = true">Delete Program</md-button>  </md-field>  <md-dialog-confirm        :md-active.sync="showDeleteDialog"        md-title="Delete this program?"        md-content="Your program will be deleted."        md-confirm-text="Continue"        md-cancel-text="Cancel"        @md-cancel="onCancel()"        @md-confirm="delete\_pr()" />  </div>  </template> |

A program szkriptje:

|  |
| --- |
| <script>  import \* as Blockly from 'blockly/core';  import \* as En from "blockly/msg/en";  import BJavascript from "blockly/javascript";  import 'blockly/blocks';  import Blocks from "@/components/parts/Blocks";  import {send\_data,devices,init,get\_data,send\_finish} from "@/components/BlocklyJS";  //import {media} from "blockly/media";  import {FireDb,FirebaseAuth,userId} from "@/firebase";  import {ref, set ,onValue,get, child,push,runTransaction } from "firebase/database";  import {add\_program} from "@/mod\_data/set\_data";  import {get\_encoding} from "@/mod\_data/get\_data";  import {delete\_program} from "@/mod\_data/del\_data";  import {encode,decoding} from "@/datas";  import {saveTextAsFile} from "@/file";  export default {      name: 'Programs',      data: () => ({        program:{},        showDialog: false,        a\_program\_name: "",        a\_program\_xml: "",        a\_program\_javascript: "",        camera:{},        devices:[],        showDeleteDialog:false,        Workspace:null,        prev:"",        js\_code:""      }),      comments:{        Blockly,      },      components:        {          Blocks        },       methods: {      myUpdateFunction(event) {},      save() {        if (this.program.program\_javascript != "") {          const userId = FirebaseAuth.currentUser.uid;          let \_ref= ref(FireDb, `/users/${userId}/rooms/${this.$route.params.rid}/programs/${this.$route.params.pid}/program\_xml`);          set(\_ref,this.program.program\_xml);          }          console.log("WELP");        },        delete\_pr()        {                     delete\_program(this.$route.params.rid,this.$route.params.pid);        },        onCancel () {                          //this.value = 'Disagreed'                      },        achange()        {                 if(this.prev==this.a\_program\_xml) return;                          const userId = FirebaseAuth.currentUser.uid;                          //console.log(this.device.mode);                          let \_ref= ref(FireDb,  `/users/${userId}/rooms/${this.$route.params.rid}/programs/${this.$route.params.pid}/program\_xml`);                          this.program.program\_xml=encode(this.a\_program\_xml,get\_encoding(this.$route.params.rid,this.$route.params.pid));                          set(\_ref,this.program.program\_xml);                          this.prev=this.a\_program\_xml;                          this.$noty.success("Saved!");         },         namechange(){                          const userId = FirebaseAuth.currentUser.uid;                          let \_ref= ref(FireDb,  `/users/${userId}/rooms/${this.$route.params.rid}/programs/${this.$route.params.pid}/program\_name`);                          set(\_ref,this.program.program\_name);                      },          auto\_compile() {        console.log("im here");        //this.program.program\_javascript = Blockly.Javascript.workspaceToCode(this.Workspace);        let xml = Blockly.Xml.workspaceToDom(this.Workspace);        console.log(xml);        if (xml != "") {          this.a\_program\_xml = Blockly.Xml.domToText(xml);          console.log(this.a\_program\_xml);          this.achange();          }      },          start()      {        //let blocklyDefault = document.getElementById("blocklyDefault");        //media: "/media/",      if(this.program.program\_xml==undefined||this.program.program\_xml==null) return;      console.log(get\_encoding(this.$route.params.rid,this.$route.params.pid));      this.a\_program\_xml=decoding(this.program.program\_xml,get\_encoding(this.$route.params.rid,this.$route.params.pid));        let workspace\_default = Blockly.Xml.textToDom(this.a\_program\_xml);      Blockly.Xml.appendDomToWorkspace(workspace\_default,this.Workspace);        this.auto\_setup();    },     duplicateprogram()    {      add\_program(this.$route.params.rid,this.program.program\_name,this.program);      this.$noty.success("Success!");      },     get\_config()                      {                        let filename="program.json";                        let l={data:this.program};                        saveTextAsFile(JSON.stringify(l),filename);}    },    mounted() {      Blockly.setLocale(En);        this.Workspace = Blockly.inject("blocklyDiv", {        toolbox: document.getElementById("toolbox"),        scrollbar: false,      });       this.Workspace.addChangeListener(() => {        this.auto\_compile(this.Workspace);      });      console.log(this.$route.params);          //localStorage.setItem('device',JSON.stringify(null));          const userId = FirebaseAuth.currentUser.uid;          const devId = this.$route.params.pid;          const room\_id=this.$route.params.rid;           onValue(ref(FireDb,  `/users/${userId}/rooms/${room\_id}/programs/${devId}`),(sn)=>{         if(sn.exists())         {           this.program=sn.val();          //this.select=this.device.mode;         }          });         onValue(ref(FireDb, `/users/${userId}/rooms/${room\_id}/devices`),(sn)=>{         if(sn.exists())         {           sn.forEach((a)=>{            this.devices.push(              {                devID:a.key,                data:a.val()              }            );           });           }           this.start();          });    },      } |

Stílusa a kinézetnek:

|  |
| --- |
| </script>  <style lang="scss" scoped>  #blocklyDiv  {    width: 800px;    height: 480px;  }  #text\_programjavascript  {  }  #text\_programxml  {    display: none;  }  </style> |

Mindegyik Blockot definiálni kell, ez alapján jelenhet meg az adott blocknak a kódja a fordítóban illetve a Sandboxban is.

Pl. A Blocks.vue fájlban találhatóak az elérhető blokkok melyek között ott vanak a gyári logikai blokkok is (IF, While, For return, String kezelő blokkok).

|  |
| --- |
| <category name="Blocks Library" colour="%{BKY\_MATH\_HUE}">        <block type="start\_room"></block>        <block type="send\_data"></block>        <block type="get\_data"></block>        <block type="send\_finish"></block>      </category> |

Ezeknek a blokkoknak a leírását egy külön Javascript file foglalkozik melyben csak megvannak hívva és külön fájlban vannak pontosan meghatározva az egyszerűség kedvéért.

|  |
| --- |
| import \* as Blockly from 'blockly/core';  import \* as En from "blockly/msg/en";    const send\_data= require("./send\_data");  const devices= require("./devices");  const init= require("./init");  const get\_data= require("./get\_data");  const send\_finish= require("./send\_finish");  const start\_room = require("./start\_room");  import Blocks from "@/components/parts/Blocks";      export {     send\_data,     devices,     init,     get\_data,     send\_finish,     start\_room  } |

get\_data fájl tartalma:.

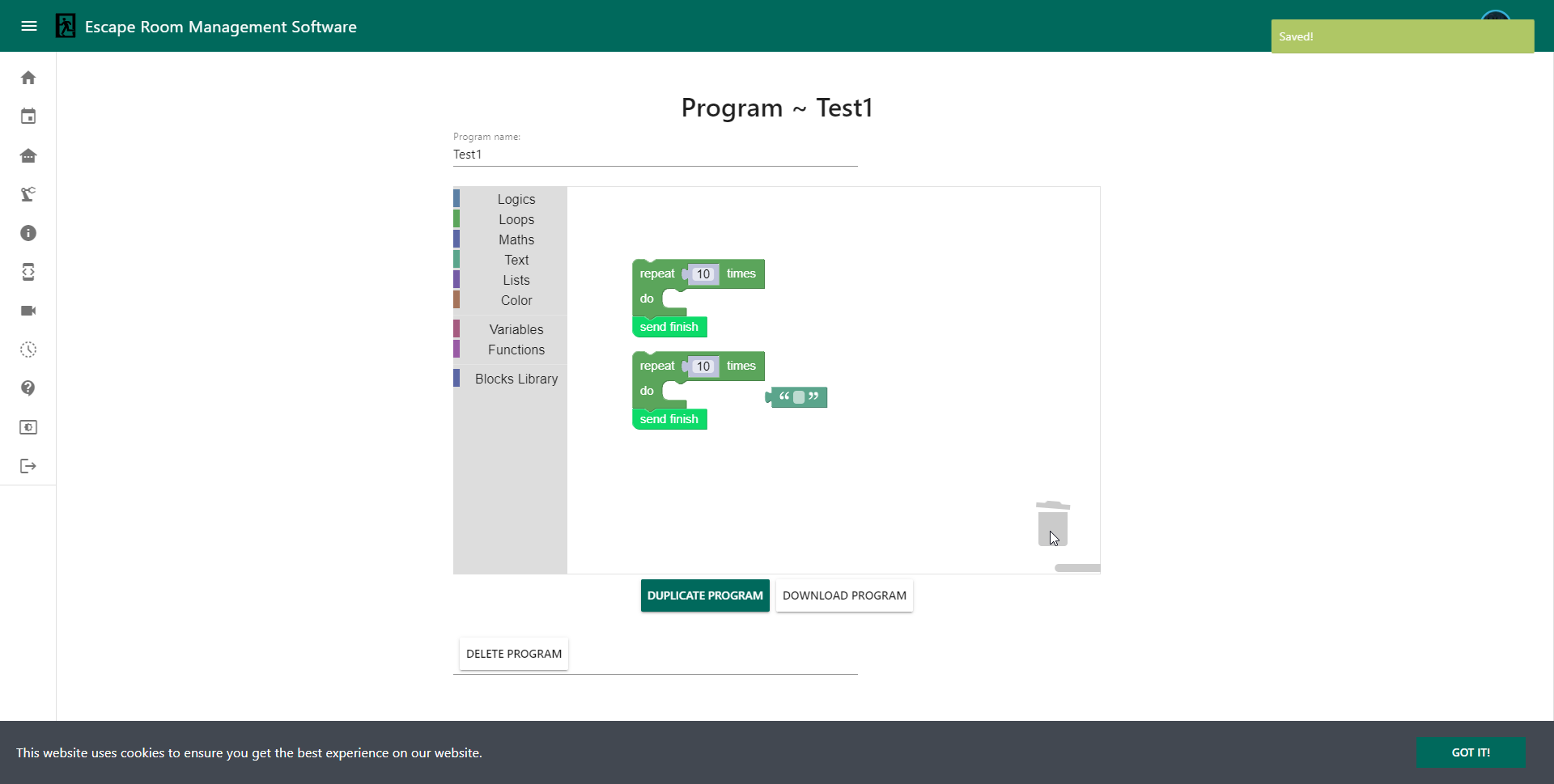
|  |
| --- |
| import \* as Blockly from 'blockly/core';  import "blockly/javascript";    Blockly.defineBlocksWithJsonArray([     {         "type": "get\_data",         "message0": "get device %1 %2",         "args0": [             {                 "type": "input\_value",                 "name": "device",                 "check": ["String"]             },             {                 "type": "input\_value",                 "name": "mode",                 "check": ["String"]             },         ],         "message1": "Then, do %1",         "args1": [             {                 "type": "input\_statement",                 "name": "DO0"             },         ],         "previousStatement":true,         "nextStatement": true,         "inputsInline": true,             "colour": '#0ddb69',         "tooltip": "",         "helpUrl": "",       },  ]);  Blockly.JavaScript['get\_data'] = function(block) {     var device = Blockly.JavaScript.statementToCode(block, 'device');     var mode = Blockly.JavaScript.statementToCode(block, 'mode');       var code = `get\_data(${device},${mode});`;     return [code, Blockly.JavaScript.ORDER\_FUNCTION\_CALL]; |

Amit a rendszeren keresztül meg kap a vue.js.

|  |
| --- |
|  |

A program átadja az xml, és a javascript változatát a programnak, ezt a Blockly felhasználja és a Vue beilleszti a két kijelölt textarea html tag-be. Minden egyes változatatásnál a felületen a Blockly.JS nevű könyvtár legenerálja az xml, valamint a Javascript változatot az aktuális beállításhoz képest.  
Majd a Save gombra kattintva lementi az aktuális állását javascript, és az xml változatnak az Firebase könyvtár segítségével.

A program nem támogatja a kódok kézi hozzáadását.



*ábra 7 Program szerkesztő*

És az alábbi szkript amit a program generált base64-be lekódolt xml-struktúra.

Ezt feldolgozza a következő script ami az api.php fájlban található.

* + 1. **Programok futattása:**

A programok futattása az ún lobby.vue fileban található meg. Ez gyakorlatilag egy Sandbox-ot hoz létre az aktuális futó szkriptnek, és ebben fut a leadott program is. Először is létrehozza az aktuális „futamot”, melynél elmenti a programot, a csapat id-t, a kezdeti dátumot, és NULL értéket ad a befejezési dátumnak (innen tudjuk, hogy aktív a futam).

A programot betöltés után a mounted függvény megnyitja és a Acorn fordító Sandbox-ába téve futtatja le. Szükséges megemlíteni, hogy a homokozóban 10 000 maximális lépés van meghatározva.

A programnak van egy belső konzolja amivel „kommunikál” az Axios keretrendszeren keresztül a perifériákkal, és ezek segítségével állítja át az esp-k tulajdonságait, pl.

|  |
| --- |
| beforeMount()          {              this.status=status\_run();              localStorage.setItem("room\_id",this.$route.params.rid);              localStorage.setItem("user\_id",FirebaseAuth.uid);          },          mounted()          {             this.get\_data();            // console.log(this.room);              console.log(this.sounds)             Blockly.setLocale(En);             const room\_id=this.$route.params.rid;             localStorage.setItem("roomID",room\_id);              this.cameras=get\_data\_fromroomdb(room\_id,"cameras");              this.devices=get\_data\_fromroomdb(room\_id,"devices");              this.Workspace = Blockly.inject("blocklyDiv", {                  toolbox: document.getElementById("toolbox"),                  scrollbar: false});             Blockly.JavaScript.INFINITE\_LOOP\_TRAP = null;             this.a\_xml=decoding(this.program.program\_xml,get\_encoding(this.$route.params.rid));             console.log(this.a\_xml);             let workspace\_default = Blockly.Xml.textToDom(this.a\_xml);             Blockly.Xml.appendDomToWorkspace(workspace\_default,this.Workspace);             this.a\_js = Blockly.Javascript.workspaceToCode(this.Workspace);              }, |

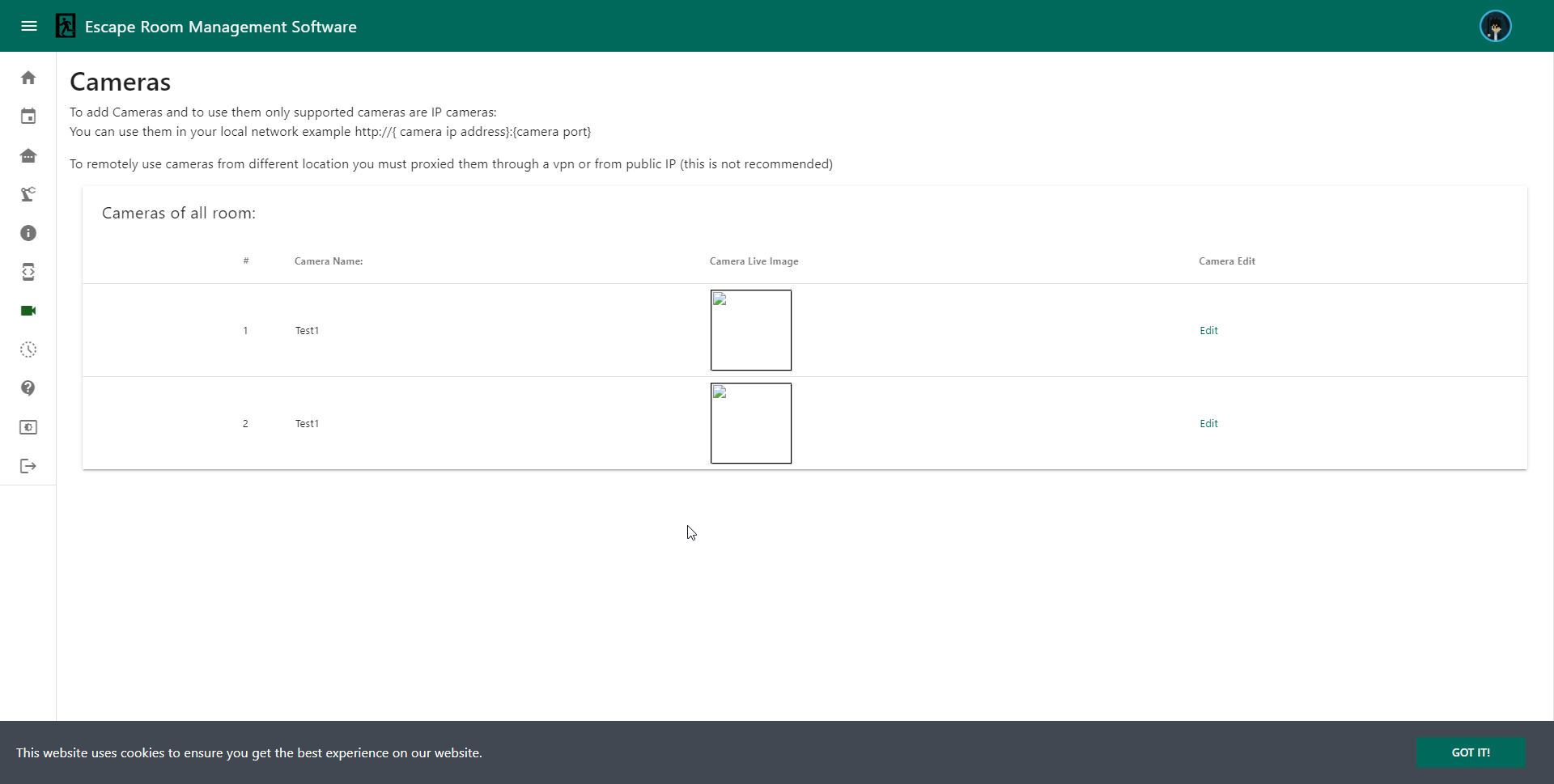
S ezt használja ki a Blockly-ba írt program is.

|  |
| --- |
| import \* as Blockly from 'blockly/core';  import "blockly/javascript";  import axios from 'axios';  Blockly.defineBlocksWithJsonArray([      {          "type": "send\_data",          "message0": "set device %1, mode: %2, status: %3",          "args0": [              {                  "type": "field\_variable",                  "name": "device",                  "check": ["String"]              },              {                  "type": "field\_dropdown",                  "name": "mode",                  "variableTypes": [""],                  "options": [                      ["Relay", "relay"],                      ["RFID reader", "rfid"],                      ["Input", "input"],                      ["OLED Display", "oled"],                  ]              },              {                  "type": "input\_value",                  "name": "value",                  "check": ["String"]              }          ],          "inputsInline": true,          "previousStatement": true,          "nextStatement": true,          "colour": 160,          "tooltip": "",          "helpUrl": "",        },  ]);  Blockly.JavaScript['send\_data'] = function(block) {      var device = Blockly.JavaScript.statementToCode(block, 'device');      var mode = Blockly.JavaScript.statementToCode(block, 'mode');      var value = Blockly.JavaScript.statementToCode(block, 'value');      var code = `set\_data(${device},${mode},${value});\n`;      return  [code, Blockly.JavaScript.ORDER\_FUNCTION\_CALL];  };  function set\_data(device,mode,value)      {            const res = axios.patch(build\_link(), { mode: mode,status:value });      }      function build\_link( device\_id="")  {      let user=localStorage.getItem("user\_id");      let room=localStorage.getItem("room\_id");      return "https://escaperoom-b4ae9-default-rtdb.europe-west1.firebasedatabase.app/users/"+user+"/rooms/"+room+"/devices/"+device\_id+".json";  }  export {      set\_data  } |

* + 1. **Kamerák kezelése:**

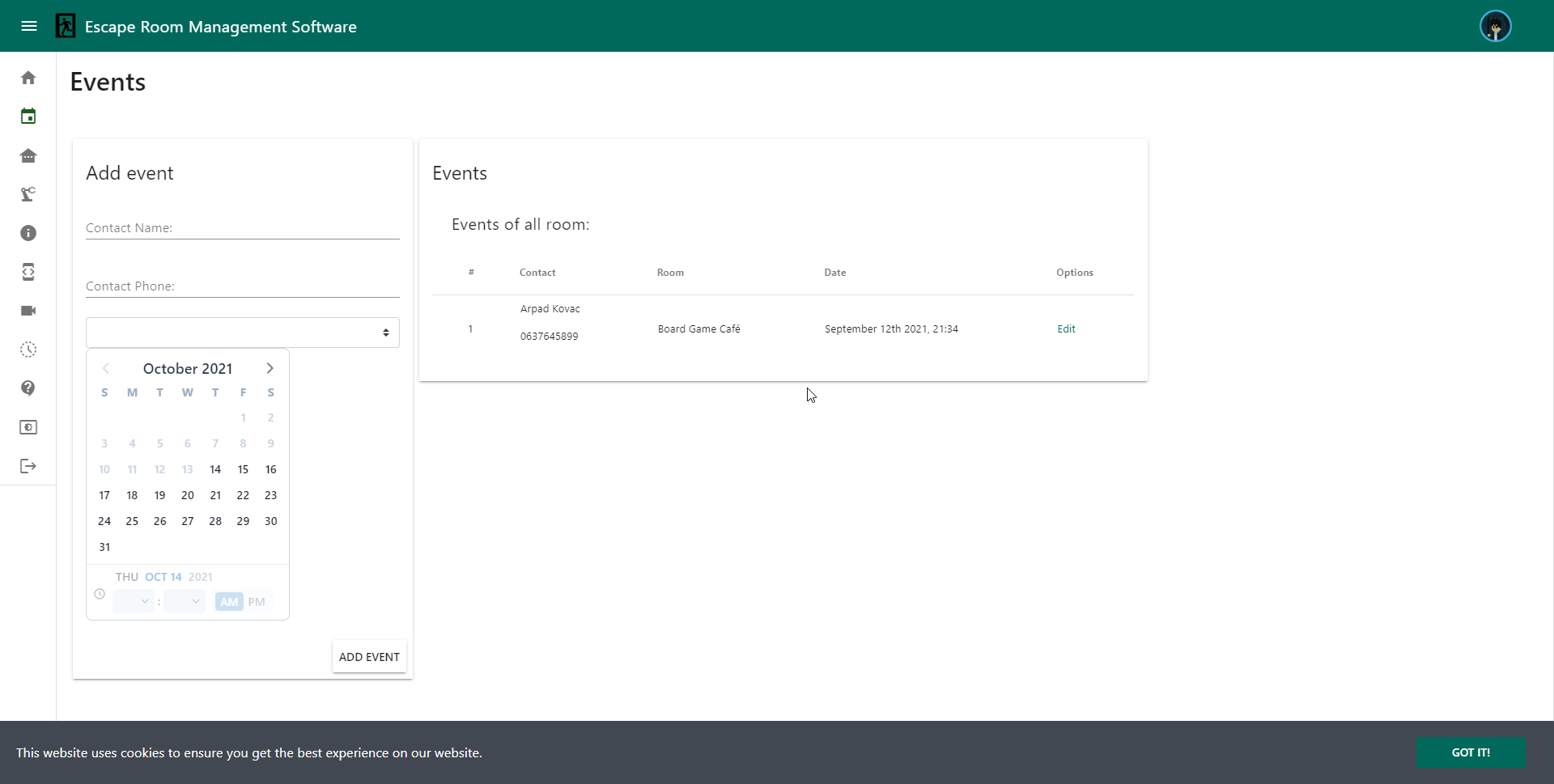
Mint minden szabaduló szobában szükséges megfigyelő eszközöket elhelyezni a játékosok tippjeinek, vagy megakadásának megsegítésére. Ezért szükséges volt a programba beültetni egy Kamera kezelő modult, mely az aktuális szobára képes IP webkameráról képet adni.

Ezt úgy érjük el, hogy az esp hálózatba csatoljuk a webkamerákat is, és megadjuk azok streaming címét, ezáltal a program ha rákattintunk az adott kamerára képes annak képét megjeleníteni.



* + 1. **Az eventek kezelése:**

A szoftver támogatja események hozzáadását, szobánként ezáltal nyomon követhetővé téve az



* + 1. **Az eszközök kezelése:**

Az eszközök kezelését két fő részre lehet bontani, van az automatikus kezelés (program általi), és van kézi vezérlés.

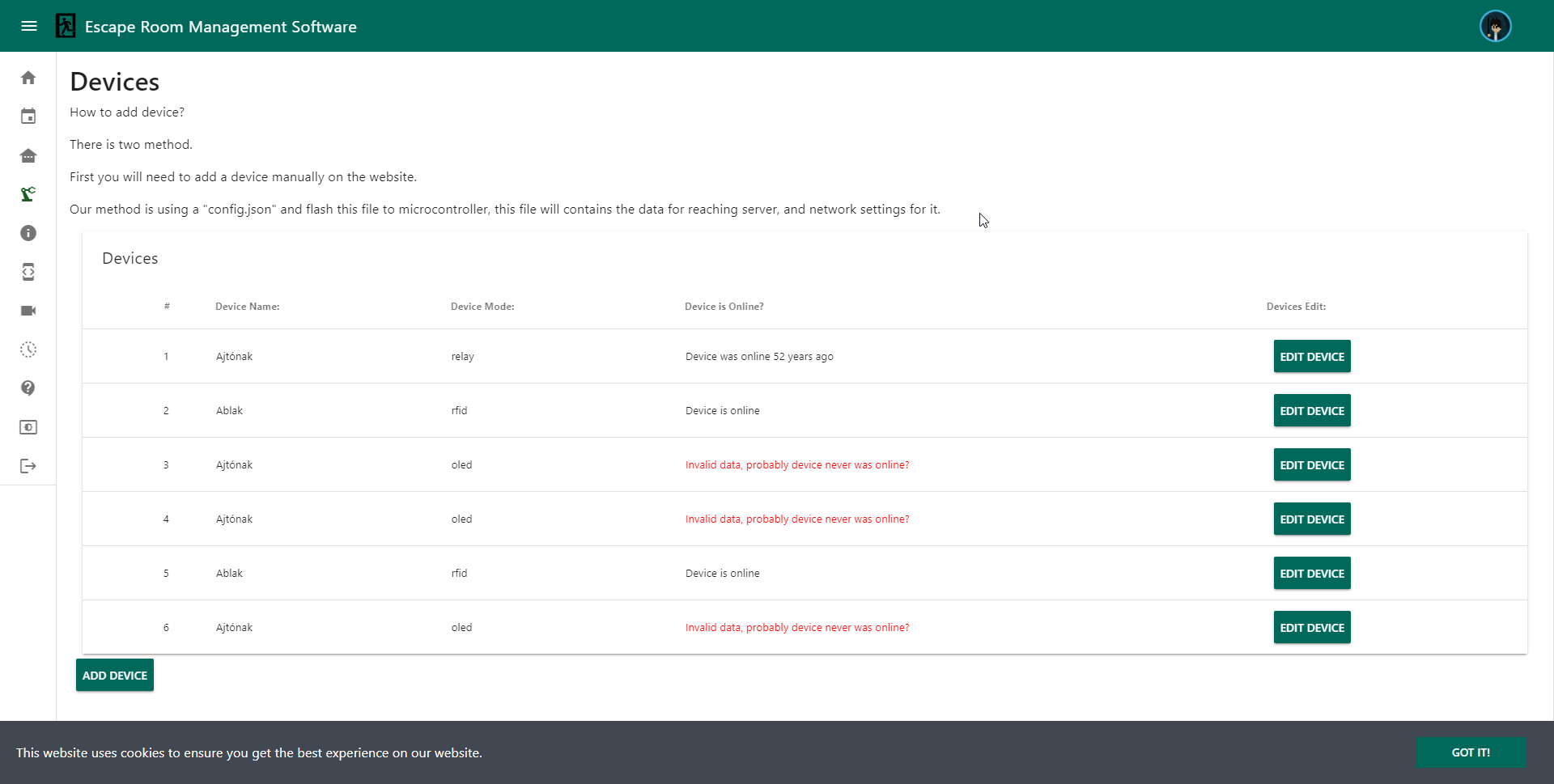
Az alábbi részben a kézi vezérlésről lesz szó.

A vue rész ami kezeli az eszközöket:.

A template-je a weboldalnak:.

|  |
| --- |
| <template>  <div >  <h2>Devices</h2>  <p>How to add device?</p>  <p>There is two method.</p>  <p>First you will need to add a device manually on the website.</p>  <p>Our method is using a "config.json" and flash this file to microcontroller, this file will contains the data for reaching server,  and network settings for it.</p>  <div class="section" v-if="devices.length>0">      <md-table  md-card>        <md-table-toolbar>          <h1 class="md-title">Devices</h1>        </md-table-toolbar>        <md-table-row>          <md-table-head md-numeric>#</md-table-head>          <md-table-head>Device Name:</md-table-head>          <md-table-head>Device Mode:</md-table-head>          <md-table-head>Device is Online?</md-table-head>          <md-table-head>Devices Edit:</md-table-head>        </md-table-row>          <md-table-row v-for="(row,index) in devices" :key="index">          <md-table-cell md-numeric>{{index+1}}</md-table-cell>          <md-table-cell>{{row.data.device\_name}}</md-table-cell>          <md-table-cell>{{row.data.mode}}</md-table-cell>          <md-table-cell><activedevice :lastonline="row.data.lastonline"/></md-table-cell>          <md-table-cell><md-button class="md-raised md-primary" @click="edit(`/room/${row.room\_id}/device/${row.dev\_id}`)">Edit Device</md-button></md-table-cell>      </md-table-row>          </md-table>        <md-button class="md-raised md-primary" @click="showDDialog = true">Add Device </md-button>    </div>       <md-empty-state v-else        md-icon="precision\_manufacturing"        md-label="You do not have devices"        md-description="You do not have registered device. You can create device inside the Room. :/">      </md-empty-state>  </div>  </template> |

|  |
| --- |
| <script>  import Activedevice from "@/components/parts/Activedevice";  import ElapsedTime from "@/components/ElapsedTime";  import router from "@/router";  import {FireDb,FirebaseAuth,userId} from "@/firebase";  import {ref, set ,onValue,get, child,push,runTransaction } from "firebase/database";  import {BIconCheck2,BIconPlus} from 'bootstrap-icons-vue'  import {get\_data\_from\_allroomdb} from "@/mod\_data/get\_data";  export default {        name: 'Devices',      data: () => ({        showDialog: false,        scan:null,        found\_devices:[],        showDDialog:false        }),      components:{        Activedevice      },      beforeMount()      {         const userId = FirebaseAuth.currentUser.uid;            /\*              még nincs implementálva a böngészőkben OwO              this.scan = navigator.bluetooth.requestLEScan({"acceptAllAdvertisements":true});              navigator.bluetooth.addEventListener('advertisementreceived',this.findADevice(event));               //          \*/      },      methods:      {/\*          findADevice(event)          {              this.found\_devices.push(                  {                      name:event.device\_name,                      device\_id:event.device.id                  }              )          },          stopScan() {              console.log('Stopping scan...');              this.scan.stop();              console.log('Stopped.  scan.active = ' + this.scan.active);              },          search\_devices()          {setTimeout(this.stopScan, 10000);},\*/  edit(l)  {    router.push(l);  },          },          computed:          {            devices()            {              return get\_data\_from\_allroomdb("devices");            }          }        }  /\*  <h2>Searching for Devices</h2>  <md-list v-if="!found\_devices || !found\_devices.length">      <md-list-item>      </md-list-item>  </md-list>  <md-table  md-card>        <md-table-toolbar>          <h1 class="md-title">Searching for Devices</h1>        </md-table-toolbar>        <md-table-row>          <md-table-head md-numeric>#</md-table-head>          <md-table-head>Device Name:</md-table-head>          <md-table-head>Device Mode:</md-table-head>          <md-table-head>Device is Online?</md-table-head>          <md-table-head>Devices Edit:</md-table-head>        </md-table-row>   <md-table-row v-for="(row,index) in found\_devices" :key="index">          <md-table-cell md-numeric>{{index+1}}</md-table-cell>          <md-table-cell>{{row.device\_name}}</md-table-cell>          <md-table-cell>{{row.mode}}</md-table-cell>          <md-table-cell><activedevice :lastonline="row.lastonline"/></md-table-cell>          <md-table-cell><md-button class="md-raised md-primary" @click="showADDialog = true">Add Device </md-button></md-table-cell>      </md-table-row>   </md-table>   <md-button class="md-raised md-primary">Add device</md-button>  \*/  </script> |



*ábra 8 Saját eszközök áttekintése*

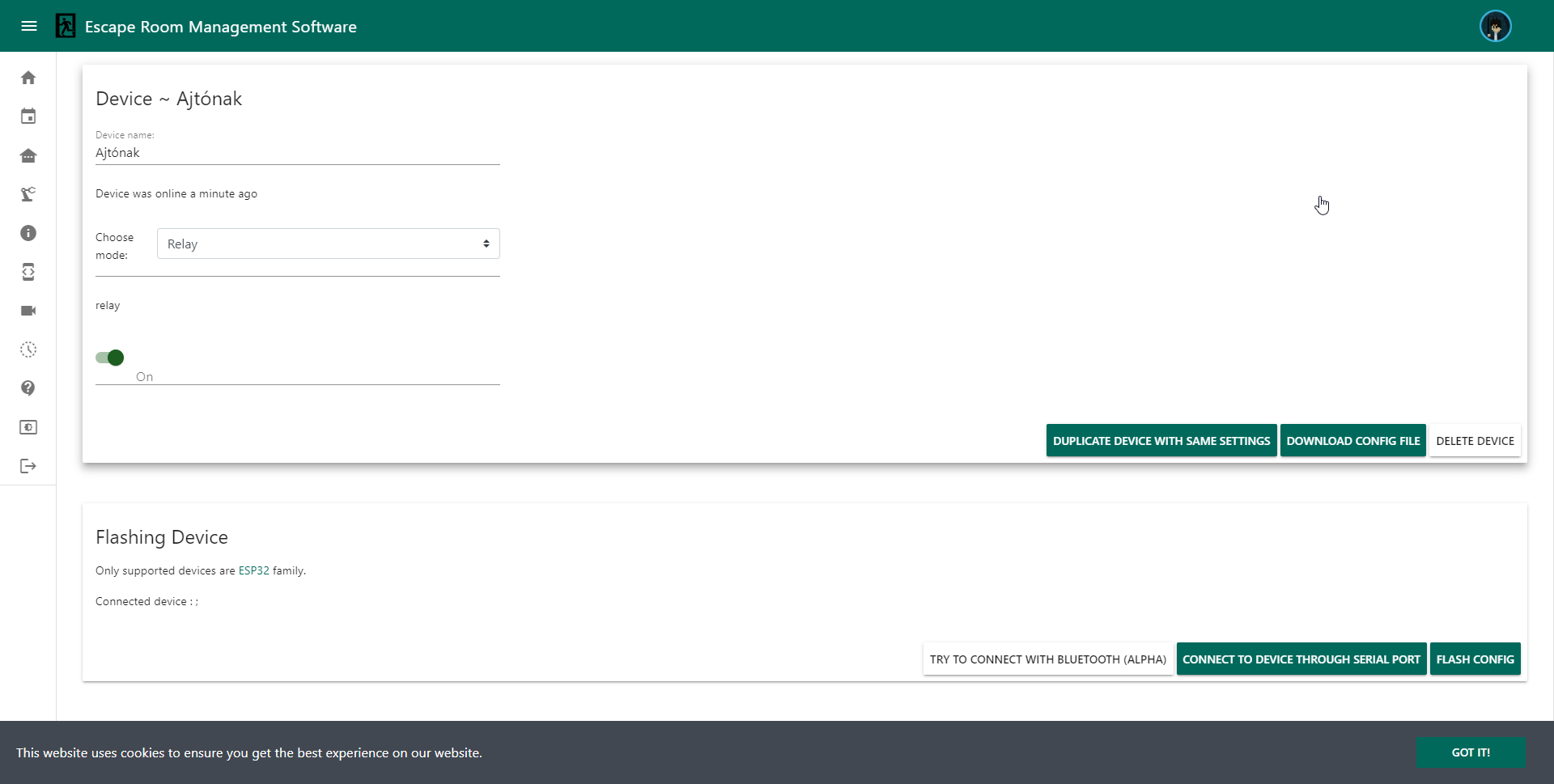
Valamint van lehetőség az eszközöknek a saját konfigurációjára is.

A template skeletonja:

|  |
| --- |
| <template>  <div >  <div class="section">  <md-card md-with-hover >           <md-card-header>           <div class="md-title" >Device ~ {{device.device\_name}}</div>          </md-card-header>          <md-card-content>            <md-field>              <label for="name">Device name:</label>              <md-input id="name" @change="namechange" v-model="device.device\_name"></md-input>            </md-field>      <p><Deviceactivity :lastonline="device.last\_online" /></p>        <md-field>        <p >Choose mode:</p>     <b-select id="select" @change="achange" v-model="device.mode">       <b-select-option v-for="mode in devm" :key="mode.type" :value="mode.type">{{mode.name}}</b-select-option>     </b-select>          </md-field>         <DeviceInput :mode="device.mode" :v-model="device.status" />         </md-card-content>         <md-card-actions>        <md-button class="md-raised md-primary" @click="duplicatedevice">Duplicate device with same settings</md-button>        <md-button class="md-raised md-primary" @click="download\_file">Download Config file</md-button>        <md-button class="md-raised md-secondary" @click="showDeleteDialog = true">Delete Device</md-button>        </md-card-actions>       </md-card>     </div>     <div class="section">  <md-card md-with-hover  v-if="serial\_supported">        <md-card-header>          <div class="md-title" >Flashing Device</div>        </md-card-header>        <md-card-content>          <p>Only supported devices are  <a href="https://en.wikipedia.org/wiki/ESP32">ESP32</a> family.</p>          Connected device : {{serial.device\_name}};          <div >            <p></p>          </div>        </md-card-content>        <md-card-actions>          <md-button class="md-raised md-secondary" :v-if="bluetooth\_active" @click="bluetooth\_connect">Try to connect with Bluetooth (alpha)</md-button>          <md-button class="md-raised md-primary" @click="check\_ports\_usb">Connect to device through Serial port</md-button>          <md-button class="md-raised md-primary" @click="flash\_config">Flash config</md-button>        </md-card-actions>      </md-card>  <md-dialog-confirm        :md-active.sync="showDeleteDialog"        md-title="Delete this device?"        md-content="Your device settings will be deleted and cant be recovered."        md-confirm-text="Agree"        md-cancel-text="Disagree"        @md-cancel="onCancel()"        @md-confirm="adelete()" />  </div>  </div>  </template> |

A program szkriptje:

|  |
| --- |
| <script>  import {FireDb,FirebaseAuth,userId} from "@/firebase";  import {ref, set ,onValue,get, child,push,runTransaction } from "firebase/database";  import Deviceactivity from '@/components/parts/Deviceactivity';  import DeviceInput from '@/components/parts/DeviceInput';  import {check\_serial\_supported,check\_bluetooth\_supported,list\_coms} from '@/components/flash\_device';  import {devicemodes} from "@/datas";  import BluetoothTerminal from "@/bluetooth/BluetoothTerminal";  import {get\_data\_fromroomitemdb} from "@/mod\_data/get\_data";  import {add\_device} from "@/mod\_data/set\_data";  import {delete\_device} from "@/mod\_data/del\_data";  import {saveTextAsFile} from "@/file";  export default {   name: 'Device',      data: () => ({        showDialog: false,        showDeleteDialog:false,        device:{device\_name:"\_Basic",mode:"relay",status:false},        devm:devicemodes,        select:"relay",        showDDialog:false,        device\_name:"",        serial\_supported:false,        serial\_device:null,        serial\_reader:null,        connected:false,        defaultDeviceName:"mcu",        serial:{device\_name:"",terminalcontainer:""},        bluetooth\_uuid:"00001101-0000-1000-8000-00805F9B34FB",        terminal : null,        bluetooth\_active:false,        encoder:null,        decoder:null,        signals:null,        reader: null,        writer: null      }),      components:{      Deviceactivity,      DeviceInput      },      mounted()      {          console.log(this.$route.params);          //localStorage.setItem('device',JSON.stringify(null));          const userId = FirebaseAuth.currentUser.uid;          const devId = this.$route.params.did;          const room\_id=this.$route.params.rid;          localStorage.setItem("mods",JSON.stringify(this.devm));          this.serial\_supported=check\_serial\_supported();          this.bluetooth\_active=check\_bluetooth\_supported();          onValue(ref(FireDb, `/users/${userId}/rooms/${room\_id}/devices/${devId}`),(sn)=>{         if(sn.exists())         {this.device=sn.val();          this.select=this.device.mode;         }          });              if(this.device.status==null)          {              set(ref(FireDb, `/users/${userId}/rooms/${room\_id}/devices/${devId}/status`),false);          }          /\*              még nincs implementálva a böngészőkben OwO              navigator.bluetooth.addEventListener('advertisementreceived',this.findADevice(event));          \*/         //setTimeout(this.serial\_log, 3000);        if(this.serial\_supported)         {            navigator.serial.addEventListener("connect",(event)=> {this.connect\_device(event)});            navigator.serial.addEventListener("disconnect",(event)=> {this.disconnect\_device(event)});         }         console.log(navigator.bluetooth);         if(this.bluetooth\_active)         {        let serviveUuid = 0xFFE0, characteristicUuid = 0xFFE1;        this.terminal=new BluetoothTerminal(serviveUuid,characteristicUuid,'\n','\n');      }      this.encoder = new TextEncoder();      this.decoder = new TextDecoder();        },      methods:      {          adelete()          {            delete\_device(this.$route.params.rid,this.$route.params.did);          },                  onCancel () {                          //this.value = 'Disagreed'                      },                       get\_type()                      {                          let b=[];                          b=JSON.parse(localStorage.getItem("mods"));                          let l="";                          console.log(b);                          if(this.device.mode==null) return "text";                          if(!b||b==null) return "text"; //to b or not 2[b]                            b.forEach((k)=>{                              if(this.device.mode===k.type)                              {                                  l=k.control;                                  return;                              }});                          //this.device.mode                          return l;                      },                       get\_readonly()                      {                          let b=[];                          b=JSON.parse(localStorage.getItem("mods"));                          let l="";                          if(this.device.mode==null) return "false";                          if(!b||b==null) return "false";                          console.log(b);                            b.forEach((k)=>{                              if(this.device.mode===k.type)                              {                                  l=k.readonly;                                   return;                              }                          });                          //this.device.mode                          return l;                      },                      async check\_ports()                      {                        let config\_for\_join={ baudRate: 9600 };                        const filters=[                          { usbVendorId: 4292, usbProductId: 60000 }                        ]                        this.serial\_device = await navigator.serial.requestPort({ filters});                        //this.serial\_device.open(config\_for\_join);                        console.log( this.serial\_device);                        console.log(this.serial\_device.getInfo());                        await this.serial\_device.open(config\_for\_join);                        //this.serial\_reader=this.serial\_device.readable.getReader();                        this.serial\_log();                      },                      async check\_ports\_usb()                      {                        let config\_for\_join={ baudRate: 115200 };                        const filters=[                          { usbVendorId: 4292, usbProductId: 60000 }                        ]                        this.serial\_device = await navigator.serial.requestPort({ filters});                        //this.serial\_device.open(config\_for\_join);                        console.log( this.serial\_device);                        console.log(this.serial\_device.getInfo());                        await this.serial\_device.open(config\_for\_join);                        this.writer = this.serial\_device.writable.getWriter();                        this.reader = this.serial\_device.readable.getReader();                        //this.serial\_reader=this.serial\_device.readable.getReader();                        this.serial\_log();                        },                        flash\_config()                      {                      },                      achange()                      {                          const userId = FirebaseAuth.currentUser.uid;                          console.log(this.device.mode);                          let \_ref= ref(FireDb, `/users/${userId}/rooms/${this.$route.params.rid}/devices/${this.$route.params.did}/mode`);                          set(\_ref,this.device.mode);                          \_ref= ref(FireDb, `/users/${userId}/rooms/${this.$route.params.rid}/devices/${this.$route.params.did}/status`);                          set(\_ref,false);                      },                      namechange()                      {                          const userId = FirebaseAuth.currentUser.uid;                          console.log(this.device.mode);                          let \_ref= ref(FireDb, `/users/${userId}/rooms/${this.$route.params.rid}/devices/${this.$route.params.did}/device\_name`);                          set(\_ref,this.device.device\_name);                      },                      duplicatedevice()                      {                        add\_device(this.$route.params.rid,this.device.device\_name,this.device);                      },                      connect\_device()                      {                        if(this.connected) return;                          this.$noty.success("Device connected!");                          this.check\_ports();                          this.connected=true;                      },                      disconnect\_device()                      {                        if(!this.connected) return;                        this.$noty.success("Device disconnected!");                        this.connected=false;                      },                      get\_config()                      {                        const userId = FirebaseAuth.currentUser.uid;                          let l={                        data:this.device,wifiname:get\_data\_fromroomitemdb(this.$route.params.rid,"wifi\_name"),                        wifipassword:get\_data\_fromroomitemdb(this.$route.params.rid,"wifi\_password"),                        device\_id:this.$route.params.did,                        room:this.$route.params.rid,                        user:userId,                        user\_email:FirebaseAuth.currentUser.user\_email,                        device\_mode:this.device.mode,                        database\_url:"https://escaperoom-b4ae9-default-rtdb.europe-west1.firebasedatabase.app",                        api\_key:"Rko9QOoaBDbXAndM0UGhKbZBFFZWFGsl6KiUNnGt",                        mod:"setup",                        accessToken:FirebaseAuth.currentUser.accessToken                        };                        return l;                      },                      bluetooth\_connect()                      {                       /\* var serviveUuid = null;                        var characteristicUuid = null;                        navigator.bluetooth.requestDevice({acceptAllDevices: true})                          .then(device => {                                console.log(device);                          });\*/                         if(this.bluetooth\_active)                      {                        this.terminal.connect().                        then(() => {                          this.serial.device\_name = this.terminal.getDeviceName() ?                              this.terminal.getDeviceName() : this.defaultDeviceName;                          });                        }                      },                      download\_file()                      {                        let filename="config.json";                        let l=this.get\_config();                        saveTextAsFile(JSON.stringify(l),filename);                      },                      async serial\_log()                       {                         let string="";                          if( !ReadableStream.prototype.pipeTo ) {                          this.$noty.error( "Your browser doesn't support pipeTo");                          return;                        }                         /\*global TextDecoderStream, textDecoder\*/                        /\*eslint no-undef: "error"\*/                        /\*const textDecoder = new TextDecoderStream();                        const readableStreamClosed = this.serial\_device.readable.pipeTo(textDecoder.writable);                        const reader = textDecoder.readable.getReader();                          const encoder = new TextEncoderStream();                        outputDone = encoder.readable.pipeTo(port.writable);                        outputStream = encoder.writable;\*/                          // Listen to data coming from the serial device.                       /\*eslint no-constant-condition: ["error", { "checkLoops": false }]\*/                          },                    write\_to\_serial(text){                       if (this.serial\_device && this.serial\_device.writable) {                          //const value = parseInt(text);                          const bytes = new Uint8Array([text]);                          const writer = this.serial\_device.writable.getWriter();                          writer.write(bytes);                          writer.releaseLock();                         }                    },                     async write(data){                      const dataArrayBuffer = this.encoder.encode(data);                      return await this.writer.write(dataArrayBuffer);                    },                     async read() {                      try {                        const readerData = await this.reader.read();                        return this.decoder.decode(readerData.value);                      } catch (err) {                        const errorMessage = `error reading data: ${err}`;                        console.error(errorMessage);                        return errorMessage;                      }                    }                  }      }  </script> |



*ábra 9 Az eszköz manuális beállítása*

* + 1. **Az ESP32 programja:**

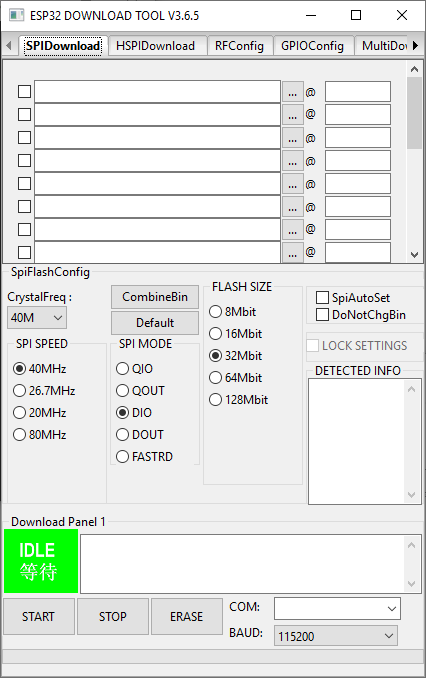
Az ESP32-nek is először szükséges a program feltöltése a programot kétféle képen lehet feltölteni az eszközre az elsőlépés, hogy a fejlesztő környezetett használjuk.

Az ESP32-t a platform io környezetben fejlesztettem, s itt ha felhasználó leklónozza a következő linket:

<https://github.com/karpad2/obudai_diplomamunka/tree/esp32_szoftver>

[S lefordítja a Platform IO-val, s telepíti.](https://github.com/karpad2/obudai_diplomamunka/tree/esp32_szoftver)

A második mód:.



*ábra 10 ESP32 gyári feltöltő szoftver*

A bin könyvtárak manuális feltöltése a gyári feltöltő szoftverrel. Mellyel feltöltjük a SPIFFS, valamint az ESP32 „factory.bin”-jét.

S ezek segítségével képes a külvilággal is kommunikálni.

Én a programozáshoz a Visual Studio Code-t használtam a Platform IO kiegészítővel.

Az eszköz flasheléséhez szükséges a platformIO particíós tábla használata:

|  |
| --- |
| # Name,   Type, SubType, Offset,  Size, Flags  nvs,      data, nvs,     0x9000,  0x5000,  otadata,  data, ota,     0xe000,  0x2000,  app0,     app,  ota\_0,   0x10000, 0x1E0000,  app1,     app,  ota\_1,   0x1F0000,0x1E0000,  spiffs,   data, spiffs,  0x3D0000,0x30000, |

NTP idő lekérdezés

Mivel szükséges, hogy az eszköz látható legyen, hogy aktív ezért szükséges volt NTP szerver használata.

Amikor a rendszer lekérdezi az időpontot ezután szükséges, még hogy a rendszer frissítse a rendszer időpontját.

A rendszer órája az UTC időpontra van beállítva, s innen tudjuk a világszinten, hogy az alkalmazás valós időben fut-e.

|  |
| --- |
| configTime(0, 0, ntpServer); //UTC idő pont      Serial.print(F("Waiting for NTP time sync: "));    time\_t nowSecs = time(nullptr);    while (nowSecs < 8 \* 3600 \* 2) {      delay(500);      Serial.print(F("."));      yield();      nowSecs = time(nullptr);    } |

Fontos volt még beállítani a DNS szervert az eszközön, sajnos erre az Espressif nem adott más lehetőséget, mint először DHCP kliensként, bejelentkezni, s azután Statikus IP címként meghatározni a DNS szervert.

Erre azért volt szükség mivel sajnos másképp, problémás volt a github oldal elérése a frissítéseknél.

|  |
| --- |
| Serial.print("Connected to ");      Serial.println(wifi\_name);      Serial.print("IP address: ");      Serial.println(WiFi.localIP());      if (MDNS.begin("ESP32-Controller")) {          Serial.println("MDNS responder started");          blinking(3);      }      WiFi.config(WiFi.localIP(), WiFi.gatewayIP(), WiFi.subnetMask(),   IPAddress(8,8,8,8)); |

Majd a csatlakozás után, szükséges volt a HTTPS kapcsolatra, muszáj volt a hitelesítő dokumentumok eltárolása is. Erre a Spiffs tárhelyet használtam fel.

Spiffs tárhely az ESP32 egy 0x3000 byte tárhelyű tár ahol, különböző nem Hard kódolt elemek vannak letárolva. Itt található a config.json, firebase\_cert.txt, és a github\_cert.txt file is.

Az eszköz aktívítását ellenőrző rész:

|  |
| --- |
| client.setCACert((char\*)firebase\_cert.c\_str());       String link=build\_link(user,room,device);       Serial.print("Firebase link:");Serial.println(link);       HTTPClient http;         if(http.begin(client,(char\*)link.c\_str())) {       dataMillis = millis();         httpCode=http.GET();       json\_response="";      if (httpCode == HTTP\_CODE\_OK) {        Serial.printf("[HTTPS] GET... code: %d\n", httpCode);        json\_response=http.getString();        Serial.println(json\_response);         http.end();        //time\_for\_response          link=build\_link(user,room,device);        if(http.begin(client,(char\*)link.c\_str())) {          String httpRequestData ="";          http.addHeader("Content-Type", "application/json");          DynamicJsonDocument timejson(2048);          timejson["lastonline"]=printLocalTime();          serializeJson(timejson,httpRequestData);          httpCode=http.PATCH(httpRequestData);          http.end(); |

* + 1. **A regisztráció menete:**

Először is az ESP32 használja a SPIFFS tárolót, és itt be van állítva egy config.json file.

Ebből a fileból lehet beállítani az eszköznek a saját felhasználó ID-jét, valamint ahhoz tartozó Szoba ID-jét, de mivel ez egy központosított rendszer, ezért több eszközről is betud jelentkezni.

Küldi a rendszer a webserial-api-n keresztül küldi az adatokat az esp32nek soros porton keresztül:

S a következő szkript fogadja:.

|  |
| --- |
| void parse\_serial(String text)  {        deserializeJson(doc, (char\*)text.c\_str());      if(doc["mode"]=="setup")      {            DynamicJsonDocument parsejson(2048);          String \_text=a\_config\_read();          deserializeJson(parsejson,\_text);          String tmp="";          tmp=doc["wifi\_name"].as<String>();          parsejson["wifi\_name"]=(tmp=="null"?parsejson["wifi\_name"]  .as<String>() : tmp);          tmp=doc["wifi\_password"].as<String>();          parsejson["wifi\_password"]=(tmp=="null"?parsejson["wifi\_password"]  .as<String>() : tmp);          tmp=doc["user"].as<String>();          parsejson["user"]=(tmp=="null"?parsejson["user"].as<String>() : tmp);          tmp=doc["room"].as<String>();          parsejson["room"]=(tmp=="null"?parsejson["room"].as<String>() : tmp);          tmp=doc["device"].as<String>();          parsejson["device"]=(tmp=="null"?parsejson["device"].as<String>() : tmp);          serializeJson(parsejson,\_text);          Serial.println(\_text);          a\_config\_write(\_text);        }  } |

Majd a config.json fáljba lementjük az adatokat.

|  |
| --- |
| void a\_config\_write(String text)  {      File file = SPIFFS.open("/config.json", FILE\_WRITE);          if(!file){              Serial.println("There was an error opening the file for writing");              return ;          }          if(file.print(text)) {              Serial.println("File was written");          }else {              Serial.println("File write failed");          }          file.close();          ESP.restart();  } |

Majd az eszközt újraindítjuk.

Ezután a rendszer válaszban megadja a lementi a wifi adatokat, illetve a felhasználónak a beállításait, lementi a config file-t.

Majd az eszközt újra indítjuk.

* + 1. **Az eszköz indulása:**

Az eszköz indulásánál először is a Spiffs filet beolvassuk majd.

|  |
| --- |
| void a\_setup\_filesystem()  {      delay(200);      Serial.println(F("Inizializing FS..."));       if(SPIFFS.begin())  Serial.println(F("done."));          else {              Serial.println(F("fail."));              SPIFFS.format();          }    } |

Ha a spiffs nem létezik létrehozzuk.

Majd folytatjuk a config file beolvássával:.

|  |
| --- |
| String a\_file\_read(String filename)  {      Serial.println("Try to read "+filename);      String configtxt="";      File file = SPIFFS.open("/"+filename);          if(!file){              Serial.println("There was an error opening the file,try to recreate it.");              file.close();              //a\_config\_write(config\_json);                return a\_file\_read(filename);              //return "";          }          else{              Serial.println("Try to open file:");              if(file.available()) configtxt=file.readString();          }            file.close();          //Serial.println(configtxt);          return configtxt;  } |

S beolvassuk az ESP32 adatait..

|  |
| --- |
| String a\_config\_read()  {      return a\_file\_read("config.json");  } |

Ha az adatok, megfelelőek a rendszer bejelentkezik a Wifi-re majd, beállítjuk az IO paramétereit.

|  |
| --- |
| void setup\_io()  {    SPI.begin();    pinMode(relaypin, OUTPUT);    pinMode(boot\_pin, INPUT);    pinMode(led, OUTPUT);    attachInterrupt(interruptpin, isr, FALLING);  } |

Elkezdi 3 másodpercenként ellenőrizni a változásokat.

Majd

* + 1. **Az eszköz bejelentkezése:**

Mint az előbb említettem a rendszernek a bejelentkezéshez szükséges a jelszó használata.

Az eszköznek 4 féle módja van előre letárolva, de lehet, hogy ez a lista még később bővül.

* Relé üzemmód.
* RFID olvasó üzemmód.
* Bemeneti üzemmód.
* OLED kijelző üzemmód.

Ebben a megvalósításban az eszköz egyszerre csak egy módszert támogat, még pedig azt amit a hálózatról kiolvas.

|  |
| --- |
| DynamicJsonDocument device\_json(2048);          deserializeJson(device\_json,(char\*)json\_response.c\_str());          String  aaa=device\_json["mode"].as<String>(); |

* + 1. **Relé üzemmód:**

Ammenyiben a bemenetre a relé üzemmód van kapcsolva, akkor az eszköz elkezdi a változóknak az elemeit értelmezi az értékét és egyet villant a led-en.

|  |
| --- |
| if(aaa=="relay")            { String a=device\_json["status"];            Serial.println(a);              bool status=a=="true"?true:false;              relay(status);          } |

A relay függvény tartalma:.

|  |
| --- |
| void relay(bool \_status)    {      digitalWrite(relaypin,\_status);      if(\_status) Serial.println("Relay: on");      else Serial.println("Relay: off");    } |

* + 1. **Bemenet üzemmód:**

A bemeneti üzemmódban a kijelölt láb (25) és a GND (föld) közötti rövidre zárásként reagál. Ebben az üzemódban egy kapcsoló reagálásra, pl. Reed relé összekapcsolására reagál.

|  |
| --- |
| else if(aaa=="input")          {              {                timejson["status"]=inputflag;                if(http.begin(client,(char\*)link.c\_str())) {                    httpRequestData ="";                    http.addHeader("Content-Type", "application/json");                    serializeJson(timejson,httpRequestData);                    httpCode=http.PATCH(httpRequestData);                    http.end();                }            }            inputflag=false;          } |

S a program reagál erre a lábra és bejelentkezik a szerverre.

|  |
| --- |
| void IRAM\_ATTR isr() {         inputflag=true;         blinking(3);      } |

Ezt fogadó php szkript:

* + 1. **Megjelenítő üzemmód**

A megjelenítő üzemmód lehető teszi, hogy egy OLED kijelzőre lehetséges legyen szöveget kiiratni.

Megjelenítése az ESP32:

|  |
| --- |
| else if(aaa=="oled")          {              display.begin(SSD1306\_SWITCHCAPVCC, 0x3C);              display.clearDisplay();              // Display Text              display.setTextSize(1);              display.setTextColor(WHITE);              display.setCursor(0,28);              String text\_oled= device\_json["status"].as<String>();              display.println((char\*)text\_oled.c\_str());              display.display();              delay(2000);              display.clearDisplay();          }          else          {            Serial.println("Fail");          } |

Fogadó szkript mely kezeli a megjelenítést:

* + 1. **RFID üzemmód**

A szoftver képes RFID kártya olvasására és jelezni a szoftver számára.

Az esp32 szkript:

|  |
| --- |
| else if(aaa=="rfid")          {                Serial.print(F("PICC type: "));                MFRC522::PICC\_Type piccType = rfid.PICC\_GetType(rfid.uid.sak);                Serial.println(rfid.PICC\_GetTypeName(piccType));                if (piccType != MFRC522::PICC\_TYPE\_MIFARE\_MINI &&                    piccType != MFRC522::PICC\_TYPE\_MIFARE\_1K &&                    piccType != MFRC522::PICC\_TYPE\_MIFARE\_4K) {                    Serial.println(F("Your tag is not of type MIFARE Classic."));                    return;                }                for (byte i = 0; i < 4; i++) nuidPICC[i] = rfid.uid.uidByte[i];                      Serial.println(F("The NUID tag is:"));                Serial.print(F("In hex: "));                readed\_rfid\_code="";                      for(byte i=0;i<rfid.uid.size;i++)                {                    readed\_rfid\_code+=String(rfid.uid.uidByte[i],HEX);                }              if(readed\_rfid\_code=="")              {                Serial.println("Not readed card");              }              else              {                timejson["status"]=readed\_rfid\_code;                if(http.begin(client,(char\*)link.c\_str())) {                    httpRequestData ="";                    http.addHeader("Content-Type", "application/json");                      serializeJson(timejson,httpRequestData);                    httpCode=http.PATCH(httpRequestData);                    http.end();                        }                        readed\_rfid\_code="";              }            } |

Mint látható az eszköz elküldi a kártya azonosító számát és beírja a Firebase status mezőjébe az értéket.

* + 1. **Raspberry PI beállítása:.**

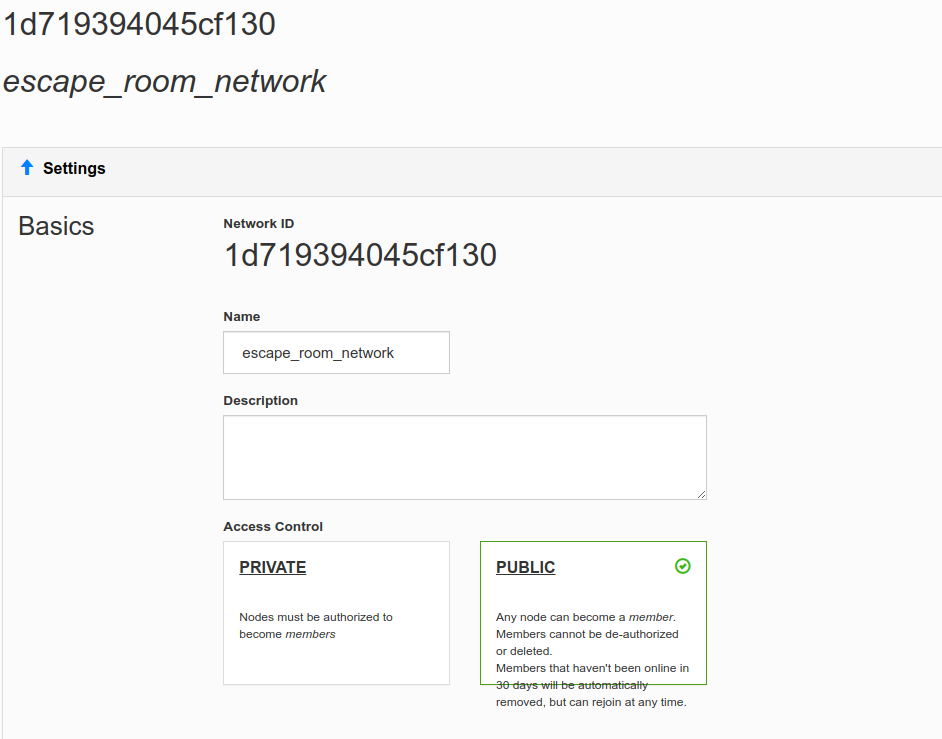
Zerotier alkalmazást feltelepítve beállítottam, hogy induláskor először lecsatlakozon a hálózatról, majd visszacsatlakozon azért, mert a képfájl ami a github tárolóban fent van, ne jelentsen ütközést a legelső indulásnál.

|  |
| --- |
| zerotier-cli leave 1d719394045cf130  zerotier-cli join 1d719394045cf130 |

Rasp Ap nyílt hozzáférésű szoftvert felraktam Manjaro operációs rendszerre. Majd beállítottam, hogy a Zerotier hálózatra adjon hozzáférést a beállított SSID, és wifi jelszó segítségével.

|  |
| --- |
| SSID: TP-Link  Jelszó: asdfghjkl123# |

Majd felállítottam egy Webmin szolgáltatást is a rendszeren, ahol átírányítottam a kéréseket a VPN szolgáltatásra. Szükséges megemlíteni, hogy ha azt akarjuk, hogy a felhasználók használhassák a kamera szolgáltatást távolról ezért szükséges használni a VPN szolgáltatást.

* + 1. **A szoftver futattása Raspberry PI-n:.**

*ábra SEQ ábra \\* ARABIC 11 Zerotier kezelő felülete*

Először is szükséges leklónozni a Github Repository tárolót.

|  |
| --- |
| git clone https://github.com/karpad2/obudai\_diplomamunka |

Ennek végeztével szükséges felrakni, manjaro operációs rendszerre a yay nevezetű szoftvert.

|  |
| --- |
| sudo pacman -Sy yay |

Majd ha ez a program feltelepült a következő szkriptet futtatjuk le:

|  |
| --- |
| yay -S node  yay -S composer |

Belépünk a repositoryba mappába.

A szükséges Javascript függőségeket feltelepítjük.

|  |
| --- |
| npm install npm run serve |

Majd elkészítjük az optimalizált verziót a futattásra.

Vagy elnavigálunk a Firebase által hostingolt linkre és bejelentkezünk :

<https://escaperoom-b4ae9.web.app/>

S a kameráknál engedélyezzük a HTTP forrás engedélyezését.

1. **Összefoglalás**

Úgy vélem, hogy a szakdolgozatban elért célokat teljesítettük. Egy olyan rendszert sikerül megvalósítani, amely hardveres modulok mellett tartalmaz egy klienst és egy web alkalmazást. A kifejlesztett modulok tökéletesen működnek.

Az ESP32 és a webalkalmazás közötti kommunikáció megfelelő müködést bizosít. A vezérlés kiegészítését jelentő, a Raspberry-n futó szerver és a web alkalmazás is olyan lehetőségekkel bővült, amelyeket a fejlesztés elején kitűztem:

* a vezérlés mobil eszközről is megvalósulhat;
* Programokat lehet készíteni, valamint futattni;
* Az eszközök állapota nyomon követhető a menüben is;

A rendszer úgy lett kialakítva, hogy a jövőben bármikor tovább lehessen a fejleszteni. Erre a felhasználók javaslatokat is tehetnek a Githubon[5].

A alkalmazást lehetne tovább fejleszteni Dockerba MongoDB-vel, de mivel a Docker ARM támogatottsága még elégé limitált, ezért ezt a célt még nem sikerült megvalósítani, tehát egy „kattintással elindítani”.

1. **Idegen nyelvű tartalmi összefoglaló**

The objective of this thesis is to design an escape room management software. A complete system of different types of hardware and software is applied. The thesis contains a web application, and client.

The ESP32 and the webapplication communication are working. The control between the Raspberry and the devices is excluded by different features:

* Control from a mobile device
* Can be programmed from a website
* Device status can be monitored from the menu.

The system contains a support tab, where users can leave issues, and ideas for upgrade, and they can leave suggestions on Github.

The application can be integrated with the Docker using only MongoDB, but the Docker ARM is quite limited, as it cannot be started with a simple application.

1. **Irodalomjegyzék**

[1] Szabaduló szoba https://hu.wikipedia.org/wiki/Szabaduló\_játék

[2] [https://nodejs.org](https://nodejs.org/), Node.JS alapok

[3] [www.inf.u-szeged.hu/miszak/utmutato-](http://www.inf.u-szeged.hu/miszak/utmutato-) Arduino – kezdő lépések

[4] <https://modi-j.webnode.hu/> - A Raspberry Pi számítógép

[5] Git tároló <https://github.com/karpad2/obudai_diplomamunka>

[6] Git tároló: https://github.com/karpad2/obudai\_diplomamunka/tree/esp32\_szoftver

[6] A program logója [https://www.lastingimpressionsonline.co.uk/image/cache/catalog/Product%20Images/Section%203/BS%20Fire%20Exit/03205-500x500.png](https://www.lastingimpressionsonline.co.uk/image/cache/catalog/Product Images/Section 3/BS Fire Exit/03205-500x500.png)

Felhasznált modulok:

- VueMaterial

- VueNoty

- Axios

- CryptoJS

- lzma

- moment

- vue-cookie-law

- vue-countdown

- blockly

- webpack

1. **Mellékletek**

## Nyomtatott melléklet

**Elektronikus melléklet**

CD, mely tartalmazza a dokumentációt és összes forráskódot.

1. https://medium.com/@joespinelli\_6190/mvc-model-view-controller-ef878e2fd6f5 [↑](#footnote-ref-2)
2. https://circuits4you.com/2018/12/31/esp32-devkit-esp32-wroom-gpio-pinout/ [↑](#footnote-ref-3)
3. https://github.com/playfultechnology/arduino-rfid-MFRC522 [↑](#footnote-ref-4)
4. [↑](#footnote-ref-5)
5. https://www.rpibolt.hu/termek/096\_128x64\_oled\_grafikus\_kijelzo\_-\_monokrom\_feher.html [↑](#footnote-ref-6)